

Computer Science Department

TECHNICAL REPORT

EXPECTED PARALLEL TIME AND SEQUENTIAL
SPACE COMPLEXITY
OF GRAPH AND DIGRAPH PROBLEMS

by

John Reif*

and

Paul Spirakis

Technical Report No. 91

October 1983

NEW YORK UNIVERSITY

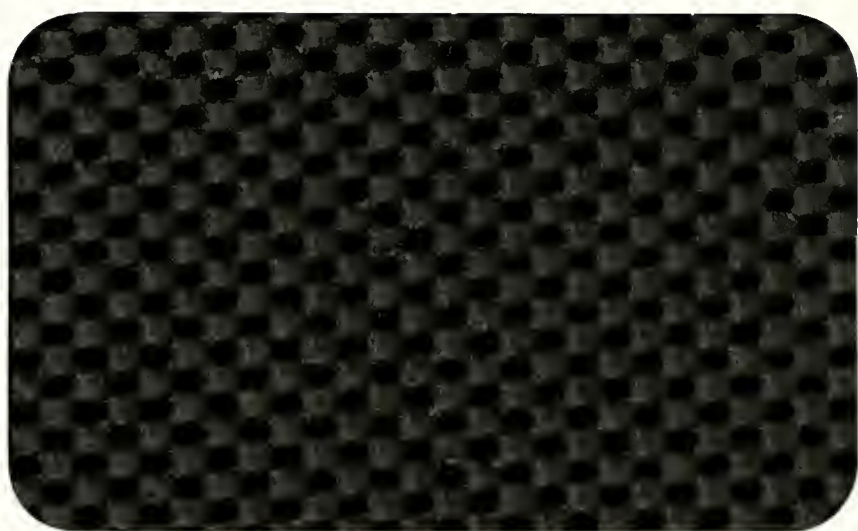


Department of Computer Science
Courant Institute of Mathematical Sciences

251 MERCER STREET, NEW YORK, N.Y. 10012

CS. TR. 91

C.1



EXPECTED PARALLEL TIME AND SEQUENTIAL
SPACE COMPLEXITY
OF GRAPH AND DIGRAPH PROBLEMS

by

John Reif*

and

Paul Spirakis

Technical Report No. 91

October 1983

* Aiken Computation Laboratory
Division of Applied Sciences
Harvard University
Cambridge, Massachusetts

This work was supported by the National Science Foundation Grants
NSF-MCS79-21024 and NSF-MCS-83-00630 and the Office of Naval
Research Contract N00014-80-C-0674.

c.1

EXPECTED PARALLEL TIME AND SEQUENTIAL SPACE COMPLEXITY
OF GRAPH AND DIGRAPH PROBLEMS

by

John Reif
Aiken Computation Laboratory
Division of Applied Sciences
Harvard University
Cambridge, Massachusetts 02138

and

Paul Spirakis
Courant Inst. of Math Sciences
Computer Science Department
New York University
New York, New York 10012

SUMMARY

This paper determines upper bounds on the *expected* time complexity for a variety of parallel algorithms for undirected and directed random graph problems. For connectivity, transitive closure, minimum spanning trees and all pairs minimum cost paths we prove the expected time to be $O(\log \log n)$ for the W-RAM (this parallel RAM machine allows resolution of write conflicts) and $O(\log n \cdot \log \log n)$ for the P-RAM (which allows simultaneous reads but not simultaneous writes). We also show that the problem of graph isomorphism has expected parallel time $O(\log \log n)$ for the W-RAM and $O(\log n)$ for the P-RAM.

Our results also improve known upper bounds on the expected space required for sequential graph algorithms. For example, we show that the problems of finding connected components transitive closure, minimum spanning trees and min cost paths have expected sequential space $O(\log n \cdot \log \log n)$ on a deterministic Turing Machine. We use a simulation of the W-RAM to get these expected sequential space bounds.

WORST CASE INPUT

	PROBLEM	P-RAM	W-RAM	
	Connectivity	$\tau = \theta(\log n \log d)$ $\pi = n^3 / \log n$ [Ja' Ja', 78] $\tau = \theta(\log^2 n)$ $\pi = n^2 / \log^2 n$ [Chin et al.81]	$\hat{\tau} = \theta(\log n)$ $\pi = n^3 \log n$ [Reif,82A]	$\tau = \theta(\log n)$ $\pi = n+m$ [Shiloah, Vishkin,81]
	K-Connectivity K Blocks	$\hat{\tau} = \theta(\log n)$ [Reif,82A] $\pi = n^3 \log n$		
UNDIRECTED	Biconnected Components (k is the number of components and d is the depth of the graph)	$\tau = \theta(\log n \log d \log k)$ $\pi = n^3 / \log n$ [Ja'Ja',78]	$\hat{\tau} = \theta(\log n)$ $\pi = n^3 \log n$ [Reif,82]	
	Min Cost Spanning Trees	$\tau = \theta(\log^2 n)$ $\pi = n^2$ [Savage, Ja'Ja',81]	$\hat{\tau} = \theta(\log n)$ $\pi = n^{\theta(1)}$ [Reif,82A]	
	All Pairs Shortest Paths	$\tau = \theta(\log^2 n)$ $\pi = n^3$ [Dekel et al.,81]		
DIRECTED	Transitive Closure and Strong Components	$\tau = \theta(\log n \log d)$ $\pi = n^3 / \log n$ [Ja'Ja',78]		
	Min Cost Spanning Trees and K-Connectivity	$\hat{\tau} = \theta(\log n)$ $\pi = n^{\theta(1)}$ [Reif,82A]		
	Isomorphism			

Table T1

Previously Known Results for Worst Case Input

π is the number of processors on worst case.

τ is parallel time for worst case input.

$\hat{\tau}$ is expected parallel time for worst case input.

n is the number of nodes and m is the number of edges

RANDOM INPUT

	PROBLEM	P-RAM	W-RAM
UNDIRECTED	Connectivity	$\bar{\tau}=O(\log n \cdot \log \log n)$ $\bar{\pi} = n^2 / \log n \cdot \log \log n$	$\bar{\tau}=O(\log \log n)$ $\bar{\pi} = n + pn^2$
	Min Cost Spanning Tree	$\bar{\tau}=O(\log n \cdot \log \log n)$ $\bar{\pi} = pn^3$	$\bar{\tau}=O(\log \log n)$ $\bar{\pi} = pn^3$
	All Pairs Shortest Paths	$\bar{\tau}=O(\log n \cdot \log \log n)$ $\bar{\pi} = n^3 / \log n$	$\bar{\tau}=O(\log \log n)$ $\bar{\pi} = n^3$
	Biconnected Components	$\bar{\tau}=O(\log n \cdot (\log \log n)^2)$ $\bar{\pi} = n^3 / \log n$	$\bar{\tau}=O((\log \log n)^2)$ $\bar{\pi} = n^3$
DIRECTED	Transitive Closure and Strong Components	$\bar{\tau}=O(\log n \cdot \log \log n)$ $\bar{\pi} = n^3 / \log n$	$\bar{\tau}=O(\log \log n)$ $\bar{\pi} = n^3$
	Minimum Cost Paths (all pairs)	$\bar{\tau}=O(\log n \cdot \log \log n)$ $\bar{\pi} = n^3 / \log n$	$\bar{\tau}=O(\log \log n)$ $\bar{\pi} = n^3$
	Minimum Spanning DAG		$\bar{\tau}=O(\log \log n)$
	Isomorphism	$\bar{\tau}=O(\log n)$ $\bar{\pi} = n^2 / \log n$	$\bar{\tau}=O(\log \log n)$ $\bar{\pi} = n^2$

Table T2

Our results for Random Input

 $\bar{\tau}$ = expected parallel time for Random Input $\bar{\pi}$ = expected number of processors for Random Input

RANDOM INPUT

	Problem	Expected Deterministic Sequential Space
UNDIRECTED	Connectivity	$O(\log n \cdot \log \log n)$
	Min Cost Spanning Tree	$O(\log n \cdot \log \log n)$
	All Pairs Shortest Paths	$O(\log n \cdot \log \log n)$
	Biconnected Components	$O(\log n \cdot (\log \log n)^2)$
DIRECTED	Strong Components, Transitive Closure	$O(\log n \cdot \log \log n)$
	All Pairs Shortest Paths	$O(\log n \cdot \log \log n)$
	Minimum Spanning DAG	$O(\log n \cdot \log \log n)$
	Isomorphism	$O(\log n \cdot \log \log n)$

Table T3

Our results for Expected Sequential Space on Random Input

1. INTRODUCTION

1.1 The Parallel Machine Models

We consider here some fundamental models of parallel computation, all of which assume the presence of an unlimited number of processors. In the first model, the P-RAM of [Wyllie, 79], [Fortune, Wyllie, 78], each processor is capable of performing arithmetic, boolean and certain read and write operations. In the P-RAM, different processors can read the same memory location at the same time. They may store information at different memory locations simultaneously, but no two processors can attempt to change the contents of the same memory cell on the same step. [Reif, 82A,B] proposed a probabilistic P-RAM where processors are capable of doing independent probabilistic choices on a fixed input (but, again, simultaneous writing at the same location is not allowed).

In another proposed machine model, the W-RAM, simultaneous access to the same memory location is allowed for both read and write operations. In the case of a simultaneous write attempt, exactly one processor succeeds but we make no assumption of which one succeeds. (See [Shiloah, Vishkin, 80] and [Goldschlager, 78] for similar models.)

A time bound for a problem in the W-RAM can be translated into a space bound for the same problem in a deterministic Turing Machine by the following Lemma.

LEMMA 1.1. *Let L be a language accepted by a $T(n)$ time-bounded W-RAM, with a polynomial number of processors. Then L is accepted by a $O(T(n) \cdot \max\{T(n), \log n\})$ space bounded deterministic Turing Machine.*

For a proof, see [Goldschlager, 78], [Cook, 80] and [Ruzzo, 82]. The result has been proved for the SIMDAG Machine which is more powerful than the W-RAM. It is also clear that

PROPOSITION: *Any parallel algorithm for the W-RAM of time T implies a parallel algorithm on the P-RAM model, of time at most $O(T \cdot \log m)$ where m is the maximum number of processors which could possibly compete for the same memory location.*

Proof. One can simulate the nondeterministic choice of write conflict resolutions by a tree of depth $\log m$ of pairwise selections to find a unique winner. □

1.2 Known Parallel Graph and Digraph Algorithms and Open Problems

We list here previous results and also provide a table (Table 1). The best known deterministic algorithm for undirected graph connectivity is of $\Theta(\log^2 n)$ time in P-RAM with $n^2/\log^2 n$ processors, [Chin, Lam, Chen, 81]. [Hirschberg, Chandra, Sarwate, 79] presented a version of the above algorithm which used $\Theta(n^2/\log n)$ processors.

Definition. Let $G = (V, E)$ be a (directed or undirected) graph and let $d(v, w)$ for $v, w \in V$ be the length of the shortest path from v to w if any, $-\infty$ otherwise. Then the *depth*, $d(G)$, of the graph G is defined to be the $\max_{v, w \in V} \{d(v, w), 1\}$. Note that for undirected graphs G with nontrivial connected components, $d(G)$ is the maximum of the diameters of the connected components of G . [Savage, Ja'Ja', 81] remarked that the [Hirschberg, et al., 79] algorithm is actually of time $\Theta(\log n \cdot \min\{\log n, d\})$ where d is the depth of the graph.

The best known deterministic algorithm for undirected connectivity in the W-RAM is due to [Shiloah, Vishkin, 81] and is of $\theta(\log n)$ time with $\theta(n+m)$ processors, where m is the number of the edges of the graph. [Shiloah, Vishkin 81] had the ingenious idea to use distributed collapsing of the height of the data structures which are candidates for connected components. Their algorithm is essentially different from that of [Hirschberg et al, 79] in the sense that the algorithm of [Hirschberg et al, 79] cannot be adjusted to run in time $\theta(\log n)$ in the stronger model W-RAM. Also, Ja'Ja', 78 has an $\theta(\log n \log d)$ time and $n^3/\log n$ processors algorithm for the P-RAM, where d is the depth of the graph. For the biconnected components, [Ja'Ja', Savage, 81] give an $\theta(\log^2 n)$ time algorithm in deterministic P-RAM with $n^3/\log n$ processors and also an $\theta(\log n \cdot \log d \cdot \log k)$ time algorithm with $mn+n^2 \log n$ processors where k is the number of components and m is the number of edges of the graph. For the minimum spanning tree problem [Ja'Ja', Savage, 81] give an $\theta(\log^2 n)$ time algorithm with n^2 processors. For all the above undirected graph problems, [Reif, 82A] gives an $\theta(\log n)$ time probabilistic algorithm for the P-RAM, requiring $n^{\theta(1)}$ processors. Note the results of [Reif, 82A] are not applicable to digraph problems.

For transitive closure and strong components in digraphs the best result [Ja'Ja', 78] for the P-RAM is of $\theta(\log n \cdot \log d)$ time with $n^3/\log n$ processors. For all pairs shortest paths in digraphs, [Dekel, Massimi, Sahni, 81] give an $\theta(\log^2 n)$ algorithm with n^3 processors. No parallel algorithm for isomorphism was known up to now.

The above discussion of previous work indicates several basic unanswered questions:

(a) Can parallel time be dropped below $\Omega(\log^2 n)$ for directed connectivity problems on the P-RAM?

(b) Can parallel time be dropped below $\Omega(\log n)$ for connectivity, bi-connectivity, minimum spanning tree, etc. on the W-RAM model?

(c) Can parallel time for undirected connectivity be dropped below $\Omega(\log^2 n)$ and *simultaneously* the number of processors be dropped below $\Omega(n^2/\log^2 n)$ on the P-RAM?

This paper gives affirmative answers to the above questions if the words "parallel time" and "number of processors" are preceded by the word "expected".

1.3 Known Sequential Space Bounds for Graph Problems

A space efficient sequential *probabilistic search* technique was first introduced by [Aleliunas, Karp, Lipton, Lovász and Rackoff, 79] to test connectivity. [Lewis and Papadimitriou, 80] observe that this technique immediately implied a space $O(S(n))$ probabilistic algorithm to simulate a symmetric nondeterministic machine with space $S(n)$. [Reif, 82A] extended the above techniques to yield $O(\log n)$ probabilistic sequential space algorithms for many undirectly graph problems. However, the above do not extend to directed graph problems, for which the problem of achieving less than $\Omega(\log^2 n)$ sequential space is still open. (The $\Omega(\log^2 n)$ space bound is implied by the theorem of [Savitch, 70].)

1.4 Average Performance of Graph Algorithms and Random Graphs

Considerable work has been done on sequential graph algorithms which are fast on the average given random input graphs. This includes the work of [Angluin, Valiant, 79], [Karp, 76], [Karp, Sipser, 81], [Schnorr, 78], [Karp, Tarjan, 80], [Reif, Spirakis, 81], [Spirakis, 81].

Almost no previous work examined the average performance of parallel graph and digraph algorithms, with the exception of [Shamir, Upfal, 82] on coloring random graphs and [Karp, Sipser, 81] on parallel matchings in random graphs. Furthermore, no previous work analyzes the expected sequential space of graph algorithms.

We analyze here the average performance of parallel algorithms for connectivity, biconnected components, strong connectivity and transitive closure, minimum cost spanning tree, minimum cost all pair shortest paths and graph isomorphism. Our analysis also gives upper bounds on the expected sequential space complexity of the above problems.

The input to our algorithms have is assumed to be a random graph of the model $G_{n,p}$ as defined in [Erdős, Rényi, 60], or a random digraph $D_{n,p}$ as defined in [Anguin and Valiant, 79].

In the Appendix we investigate the probability distribution of the depth of a random graph $G_{n,p}$ or a random digraph $D_{n,p}$ (the depth was defined in §1.2 of this paper). The mean value $\bar{d}$ (and bounds on the probability distribution) of the depth is crucial to a number of our results. We prove the following theorems:

THEOREM 1.1. *There is a constant $c_0 > 2$ such that for any probability p in the range $[0,1] - \left[\frac{c_0}{n}, \frac{2c_0}{n} - \left(\frac{c_0}{n} \right)^2 \right]$ the graph $G_{n,p}$ has average depth $\bar{d} = O(\log n)$. Furthermore, $\text{prob}\{d = O(\log n)\}$ tends to 1 as n to ∞ .*

THEOREM 1.2. *There is a constant $c_0 > 2$ such that for any probability p in the range $[0,1] - \left[1 - \sqrt{1 - \frac{c_0}{n}}, \frac{c_0}{n} \right]$, the digraph $D_{n,p}$ has average depth $\bar{d} = O(\log n)$. Furthermore, $\text{prob}\{d = O(\log n)\}$ tends to 1 as n tends to ∞ .*

We also prove that for $p \geq \frac{c}{\sqrt[3]{n}}$ (c a particular constant > 1) the depth d of $G_{n,p}$ (or $D_{n,p}$) is less than or equal to 3 with probability $\rightarrow 1$ as $n \rightarrow \infty$. Although the results $\bar{d} = O(\log n)$ can be deduced from results of [Erdős, Rényi, 59] for several values of p (see Appendix), the results for sparse graphs with edge probability $p = \theta\left(\frac{1}{n}\right)$ and the result for very dense graphs $\left(p \geq \frac{c}{\sqrt[3]{n}}\right)$ are new contributions.

1.5 Our Expected Time Results for Parallel Graph Problems

Using the above facts and some nontrivial analysis, we show that one can find the connected components of a random graph $G_{n,p}$ and the strong components and transitive closure of a random digraph $D_{n,p}$ and also the minimum spanning tree of a weighted random graph and the all-pairs shortest paths of a weighted random graph or digraph in average parallel time $O(\log \log n)$ in the W-RAM and $O(\log n \cdot \log \log n)$ in the P-RAM. The average time for biconnected components is $O((\log \log n)^2)$ for the W-RAM and $O(\log n \cdot (\log \log n)^2)$ for the P-RAM. The average time for graph isomorphism is $O(\log \log n)$ in the W-RAM and $O(\log n)$ in the P-RAM. (See also Table T.2). The results for digraph connectivity and digraph and graph all-pairs shortest paths are implied by the result that the average depth of $G_{n,p}$ or $D_{n,p}$ is $\bar{d} = O(\log n)$ (see Section 1.4).

However, our result for undirected connectivity in the W-RAM (and the result for minimum spanning tree in the W-RAM) are *not implied* by the fact that $\bar{d} = O(\log n)$. A large part of our paper is devoted to a new algorithm (and probabilistic analysis) for undirected connectivity in the W-RAM. A side-benefit of this new algorithm is a result about the expected number of processors needed for undirected connectivity in the P-RAM model, which is $n^2/(\log n \cdot \log \log n)$ for time $O(\log n \cdot \log \log n)$.

The above results answer the (open) questions (a), (b), (c) of Section 1.2 of this paper in an affirmative way, given that one concentrates in average case analysis.

We also show that the expected number of processors is $n + pn^2$ for undirected connectivity in the W-RAM (and this is linear for $pn = \theta(1)$). For directed connectivity and transitive closure, $n^3/\log n$ processors are needed for the P-RAM model and n^3 processors are expected for the W-RAM model.

For biconnected components we need $n^3/\log n$ and n^3 for the P-RAM and W-RAM, respectively. For graph isomorphism we need n^2 processors for the W-RAM and $n^2/\log n$ for the P-RAM. For the minimum spanning tree (all pairs shortest paths) we need $n^2/\log n$ ($n^3/\log n$) processors for the P-RAM and $n^2 p$ processors ($n^3 p$ processors) for the W-RAM on the average.

We remark there is generally a considerable drop in time complexity when one concentrates on the average case of the above problems rather than worst case input. For sake of comparison we list previous results at Table T1 and our time complexity results at Table T2. Note that none of the previously known algorithms for undirected connectivity implies an expected time below $\Omega(\log^2 n)$ in the P-RAM and a simultaneous drop of expected number of processors below $n^2/\log n$ even if the average depth result $\bar{d} = O(\log n)$ is used. Note also that our results for W-RAMs can be applied to SIMDAGs and other parallel computation models allowing resolution of write conflicts (see [Kucera, 81]).

1.6 Our Results for Expected Space Complexity

Our results improve known upper bounds on the expected space required for sequential graph algorithms. By using our expected parallel time results for W-RAMs and Lemma 1 we provide an upper bound of $O(\log n \cdot \log \log n)$ for the expected deterministic sequential space needed for undirected connectivity, directed (strong) connectivity and transitive closure, min cost spanning tree and directed and undirected all pairs shortest paths, and graph isomorphism. [Reif, 81A] provides $\Theta(\log n)$ probabilistic sequential space bounds for the undirected versions of all the above problems, but his method does not seem to extend to directed graph problems. Our results answer the question posed in Section 1.3 in an affirmative way when one concentrates on expected space and our methods could be viewed as indirect ways of getting results about expected sequential space.

2. EXPECTED PARALLEL TIME FOR DIGRAPH CONNECTIVITY

The following well-known algorithm for computing the transitive closure of an $n \times n$ boolean matrix A can be used for digraph connectivity and transitive closure:

```
input  A
begin
    B  $\leftarrow$  A + I
L:    B'  $\leftarrow$  B
    B  $\leftarrow$  B · B
    if B'  $\neq$  B then goto L else output B
end
```

This algorithm was shown in [Dekel, Nassimi, Sahni, 81] to be of $\Theta(\log^2 n)$ worst case time by using n^3 processors in the P-RAM model. [Ja'Ja', 78] did an efficient implementation of this algorithm showing that it could be done in $\Theta(\log n \cdot \log d)$ time on the P-RAM model and by using $n^3 / \log n$ processors, where d is the depth of the digraph D . Note that we can drop the number of processors to $n^{\log_2 7} / \log n$ by applying Strassen's algorithm for matrix multiplication. It is easy to observe, that the primitive set of operations of [Ja'Ja', 78] (i.e., the union of $O(n)$ sets of $O(n)$ integers each, from 1 to n) can be done in $O(1)$ time in the W-RAM, implying an $O(\log d)$ algorithm for strong components and transitive closure (see also [Kucera, 82]). Using our results on the average depth of $D_{n,p}$ given in the Appendix immediately imply;

THEOREM 2.1. *The average parallel time for transitive closure of directed graphs is $O(\log \log n)$ for the W-RAM model and $O(\log n \cdot \log \log n)$ for the P-RAM model. The probability that the parallel time is more than this is less than*

$n^{-\gamma}$, $\gamma > 2$. The W-RAM model uses n^3 processors and the P-RAM model uses $n^3/\log n$ processors for the stated time bounds.

[Ja'Ja', 78] gave also an algorithm for finding the strong components of a digraph in parallel time $\Theta(\log n \log d)$ in the R-RAM model, by using $n^3/\log n$ processors. Our results on the depth of $D_{n,p}$ also imply:

THEOREM 2.2. *The average parallel time for finding the strong components of a digraph in the P-RAM model is $O(\log n \cdot \log \log n)$, using $n^3/\log n$ processors. In the W-RAM model the average time is $O(\log \log n)$ using n^3 processors. The probability that the parallel time is more than this, is less than $n^{-\gamma}$ for some $\gamma > 2$.*

By using Lemma 1.1 we also get

THEOREM 2.3. *The average deterministic sequential space needed for transitive closure and strong components of digraphs and connected components of undirected graphs is $O(\log n \cdot \log \log n)$. The probability that the space needed is $O(\log n \cdot \log \log n)$ is at least $1 - n^{-\gamma}$ for some $\gamma > 2$.*

(Note that the same techniques can be applied to undirected graphs to compute the connected components by the P-RAM in $O(\log n \cdot \log \log n)$ average time. However, the number of processors used is wasteful. In the next section we give a derivation of this result which is independent of our results about depth of $G_{n,p}$ and optimizes simultaneously over expected time and expected number of processors.)

3. EXPECTED PARALLEL TIME AND PROCESSOR BOUNDS FOR CONNECTIVITY IN UNDIRECTED GRAPHS

3.0 A New Probabilistic Connectivity Algorithm

We present here a new algorithm for finding connected components of a graph on a W-RAM. Although it is related to the algorithm of [Shiloah, Vishkin, 82], it uses probabilistic choice so the processors of the W-RAM model are allowed to independently choose random bits. Also, we assume random resolution of write conflicts. If k processors attempt to write simultaneously at the same memory location, then exactly one succeeds and the probability that each particular one succeeds is $1/k$. Our algorithm for graph connectivity has $O(\log \log n)$ expected parallel time in the W-RAM which leads to an $O(\log n \cdot \log \log n)$ expected parallel time algorithm in the P-RAM. Our algorithm uses $2n+m$ processors for a graph of n vertices and m edges, hence it uses $2n + pn^2$ processors on the average. An improvement over this is $n^2/(\log n \cdot \log \log n)$ processors will be shown in section 3.2.

During the computation each vertex v has a pointer field $D(v)$ through which it points on another vertex or to itself. One can regard $v \rightarrow D(v)$ as a (directed) edge in an auxiliary graph, called the *pointers-graph*. In the probabilistic part of our algorithm, this pointers-graph will always be a collection of sets called super-vertices each of which is a rooted star of height one, with a self-loop at the roots. As the algorithm proceeds, the number of sets decreases while each individual set increases or disappears. This is caused by a "hooking" operation in which a supervertex can be hooked on another supervertex. In each phase of the algorithm each of the supervertices decide (probabilistically, depending on their size) to be either *acceptors* or *proposers*. Only proposers can hook themselves to other supervertices which must be acceptors. Partial collapsing is done in each phase so that supervertices remain being

rooted stars of height one. At the end, each remaining set is a connected component of the graph and is still a rooted star.

Let $s(t)$ be the following sequence:

$$s(1) = 2 \log^4 n$$

$$\text{and for } i \geq 1, \quad s(i+1) = \frac{s(i)^2}{2^i (1+\epsilon) \log^4 n}$$

(ϵ is a small constant, to be determined later.) We now describe the algorithm in detail.

Algorithm MERGE

Input: Graph $G = (V, E)$, an instance of the random graph $G_{n,p}$. We shall use $D_t(v) = u$ to denote that an edge from vertex $v \in V$ points to vertex u after the t^{th} phase.

Initially, at phase 1, for each $v \in V$, $\{v\}$ is a supervertex, these are the only supervertices, each supervertex is declared *large* and $D_t(v) = v$ for each vertex v .

In the following text we will use "vertex" and the assigned processor to that vertex interchangeably. We allocate a vertex processor to each vertex. Each undirected edge $\{u, v\}$ is viewed as two ordered pairs $\langle u, v \rangle, \langle v, u \rangle \in V \times V$ and we allocate one edge processor to each such pair.

The algorithm has 3 stages. Before the start of the first stage, each edge processor draws a random number between 0 and 1, uniformly. We require that $\langle u, v \rangle$ and $\langle v, u \rangle$ draw the same number, but otherwise the choices are independent. Those processors, whose number is less than $\frac{1}{3}$, participate in the first stage only. Those, whose number is between $\frac{1}{3}$ and $\frac{2}{3}$, participate in the second stage only. The rest participate in the third stage. This procedure results in partitioning the instance of the random graph $G_{n,p}$ into 3 instances of a random

graph $G_{n,p/3}$, in a way guaranteeing that edges used in one stage are not reused in another stage.

Stage 1. We apply the [Shiloah, Vishkin, 82] connectivity algorithm for $4\log\log n$ of its phases.

Stage 2. It consists of $\leq d_0 \log\log n$ phases. (d_0 is a positive constant, its exact value is determined later.) The pairs of processors associated with each edge and with stage 2, draw in the beginning of each phase of stage 2, with probability $\frac{1}{2}$. Those who succeed, are used only for that phase. Those unused, try again in the next phase. (This sampling of edges guarantees that an edge is not used in two different phases of stage 2 and hence the randomness of the input is not destroyed.)

Set $i = 1$.

Execute the following loop until $i = d_0 \log\log n$

Phase i of stage 2.

(a) Let β be a positive constant greater than 3, to be determined later. For each supervertex, construct a table of size $\frac{s(i)}{\beta \log n}$. Initially, all entries of the table are 0. Each vertex of the supervertex sets a randomly chosen cell of the table to 1 on this phase. We declare a supervertex to be *large* if all its cells are found to be 1.

(b) Let K be a constant greater than 4, whose exact value is determined later. Let the root of each large supervertex choose to be an *acceptor* with probability $\frac{\log^K n}{\alpha \cdot s(i)}$. (α is a small positive constant > 1 , to be determined later.) All other supervertices are declared *proposers*.

(c) Each proposer supervertex chooses a random edge departing from it to an acceptor (if such an edge exists) and merges into that acceptor. This is done

in constant time on the W-RAM as follows: All the processors associated with vertices of the proposer check whether their neighbors are nodes of an acceptor. If one finds such a neighbor u , it tries to hook its supervertex on $D_{i-1}(u)$. If more than one try this, then each one is allowed to succeed with equal probability by use of random resolution of simultaneous writing.

(d) (Collapsing): For all vertices v we assign $D_i(v) = D_{i-1}(D_{i-1}(v))$.

Notice that phase i takes a constant number of steps on a W-RAM. To do (a) in constant time, we must place the assigned table in contiguous memory. In addition, we will assume a distinct register, which is set to 1 at the beginning of (a). The root of each currently large proposer writes 1 into a randomly chosen memory cell of the root of the acceptor it proposes to. Then, each of the vertices of the acceptor selects randomly one of the cells of their root, and if any chosen cell has value 0, then it writes a 0 in the register of the root of the acceptor. Otherwise, they do nothing. The root of the acceptor then reads its register and declares the acceptor *large* if the register has value 1 and otherwise declares the acceptor *small*. Finally, each currently large proposer then resets its chosen memory cells to be 0, and the root of the acceptor sets its register to 1.

To do (b) in constant time on the W-RAM, we let each root of a large supervertex to make first an independent probabilistic choice and then have the processors associated with each node of the set read the result.

Stage 3. We run the (deterministic) connectivity algorithm of [Shiloah, Vishkin, 82] until the number of supervertices does not change.

Finally, output the connected components. These are given by the supervertices remaining.

3.1 Proof of Correctness of Algorithm MERGE

We prove in this section the following:

THEOREM 3.1. *Algorithm MERGE computes the connected components of $G_{n,p}$ with $p \geq c/n$ and $c \geq 6$, in expected parallel time $O(\log \log n)$ in the W-RAM. The expected number of processors used is $O(n + pn^2)$. The probability that the parallel time exceeds $O(\log \log n)$ is $o(\frac{1}{n})$. (Note that with a constant factor of time increase we can reduce the number of processors to $n + pn^2$.)*

Proof of Theorem 3.1. Let K be the number of supervertices at the end of stage 1. Let K_1 be the number of supervertices of size greater than $\frac{1}{2} \log^4 n$ and K_2 be the number of supervertices of size less than or equal to $\frac{1}{2} \log^4 n$.

Then K is at most $\frac{n}{\log^4 n}$ (since the number of supervertices drops by $\frac{1}{2}$ on each phase). Also, at least $\frac{n}{2}$ of the nodes will be in supervertices of size at least $2 \log^4 n$. If the opposite was true, then

$$(1) \quad K_1 \text{ is less than } (\frac{n}{2}) / (\frac{1}{2} \log^4 n) = n / \log^4 n$$

$$\text{and } (2) \quad K_2 \text{ is greater than } (\frac{n}{2}) / (\frac{1}{2} \log^4 n) \text{ i.e., greater than } n / \log^4 n.$$

But then $K_2 > K$. Thus we have a contradiction by the Pidgeonhole principle.

It is easy to see that:

PROPOSITION 3.1. *At the end of (a) of phase i of stage 2, each large supervertex has size at least $s(i) / (\beta \log n)$.*

We now will show

LEMMA 3.1. *If a supervertex has size $\geq s(i)$, then it is declared large with probability at least $1 - n^{-(\beta-1)}$.*

Proof. Let $p(i)$ be the number of cells of the table of the supervertex, which stay at value 0 after (a) of stage 2. We know that $s(i)$ is increasing with i and $s(1)$ is $2\log^4 n$. The probability that the particular supervertex fails to be declared large is

$$\text{prob}\{\text{There is a particular memory cell staying at value 0}\}$$

$$= (1 - 1/(s(i)/(\beta \log n)))^{s(i)} \leq e^{-\beta \log n}$$

$$\leq n^{-\beta}$$

so

$$\text{prob}\{p(i) = 0\} \leq n \cdot n^{-\beta} = n^{-(\beta-1)}$$

□

We now inductively assume that for $i \geq 1$, with probability $> 1 - n^{-2}$, "At least $\frac{n}{2^i}$ nodes are in large supervertices". The basis, $i=1$, was proved previously. Let N_i be the number of large supervertices at end of (a) of phase i .

Fact 1. For any n, p, β with $n > 0$, $1 \geq p \geq 0$ and $0 \leq \beta \leq 1$

$$(a) \quad \sum_{K=0}^{\lfloor (1-\beta)np \rfloor} \binom{n}{K} p^K (1-p)^{n-K} \leq \exp(-\beta^2 np/2)$$

and (b)
$$\sum_{K=\lceil (1+\beta)np \rceil}^n \binom{n}{K} p^K (1-p)^{n-K} \leq \exp(-\beta^2 np/3)$$

The above fact follows from the bounds of [Chernoff, 52].

LEMMA 3.2. For any $\beta' \in (0,1)$, if $N_i \geq s(i) \cdot \left(\frac{\beta-1}{\alpha}\right)$ then the number of large acceptors at the end of (b) is at least $L = (1-\beta')N_i \cdot \frac{\alpha \log^4 n}{s(i)}$ with probability $\geq 1 - n^{-2(\beta')^3}$.

Proof. By Fact 1, applying it to the Bernoulli experiment of N_i trials and success probability $r = \log^K n / (\alpha \cdot s(i))$, we get

$$\begin{aligned} \text{Prob}\{\text{number of successes} \geq L\} &\geq 1 - e^{-(\beta')^2 N_i (r/2)} \\ &\geq 1 - e^{(\beta')(3/2) \cdot \log^K n} \geq 1 - n^{-K/2 (\beta')^3} \geq 1 - n^{-4 (\beta')^3} \quad \square \end{aligned}$$

By Lemma 3.2, and Proposition 3.1 we then get:

COROLLARY 3.1. The number of nodes in acceptors at the end of phase i , step (b) stage 2 is at least $(1-\beta') N_i \left(\frac{1}{\beta}\right) \log^{K-1} n$ with the same probability as that stated in Lemma 2.

Now we get an estimate on N_i based on the inductive hypothesis: Since the product of number of large supervertices times min size is at least equal to min number of nodes in them we get:

$$\frac{N_i s(i)}{\beta_1 \log n} \geq \frac{n}{2^i} \quad \text{implying} \quad N_i \geq \frac{n \log n}{2^i s(i) \beta_1} \quad (*)$$

Definition 3.1. Let q be the probability that a random edge, not yet used at stage i , goes into an acceptor, given the inequality of Corollary 3.1.

COROLLARY 3.2. $q \geq (1-\beta') \alpha \frac{1}{\beta^2} \frac{\log^K n}{2^i s(i)}$

Proof. We notice that $q = (\text{number of nodes in acceptors})/n$ and then we can apply Corollary 1. □

Now we prove our central Lemma.

LEMMA 3.3. For any $a > 4$. Each large proposer, P , has at least one departing edge into a node of an acceptor, with probability at least $1 - n^{-a}$, $a > 4$, given the condition of Corollary 3.1.

Proof. Let $s'(i) = s(i)/\log n$. Since P is large, $|P| \geq \frac{s(i)}{\beta_1 \log n}$. Let $c' = c/3$. Since edges are not reused by the algorithm, each new edge tried is a random edge, with probability $p' \geq \frac{c}{3n} = \frac{c'}{n}$ of being there. By the Chernoff bounds on a Bernoulli experiment of $|P|$ trials and success probability $p' \cdot q$, there are at least $(1 - \epsilon)(1 - \beta') c' \frac{s'(i)}{\beta}$ departing edges from P , with probability at least $1 - e^{-\epsilon^2 c' (1/\beta) s'(i)} \geq 1 - n^{-3\epsilon^2 c' (1/\beta)}$, $\epsilon \in (0, 1)$.
Let $\lambda_1 = (1 - \beta') \frac{\alpha}{\beta^2}$, $\lambda_2 = (1 - \epsilon)(1 - \beta') \frac{c'}{\beta}$. Then, by using Corollary 2, we get

$$\text{Prob}\{\text{no edge exists from } P \text{ to an acceptor}\}$$

$$\leq (1 - q)^{(s(i)/\log n) \lambda_2}$$

$$\leq (1 - \lambda_1 \cdot (\log^K n / 2^i s(i)))^{\lambda_2 s'(i)}$$

$$\leq (1 - \lambda_1 \frac{\log^{K-1} n}{2^i s'(i)})^{\lambda_2 s'(i)} \leq \left(1 - \frac{\lambda_1 \lambda_2 \log^{K-1} n / 2^i}{\lambda_2 s'(i)}\right)^{\lambda_2 s'(i)}$$

$$\leq e^{-\lambda_1 \lambda_2 \log^{K-1} n / 2^i}$$

$$\leq e^{-\lambda_1 \lambda_2 \log^{K-2} n} \leq n^{-a}$$

where $a = (K-2)\lambda_1\lambda_2$ and we can now choose ε, β' (up to row free) and $K > 4$ to make $a > 4$. □

LEMMA 3.4. (Induction Step). At least $\frac{n}{2^{i+1}}$ of the nodes are (at the end of phase i , stage 2) in supervertices of size at least $s(i+1)$, with probability at least $1 - n^{-2}$.

Proof. We condition on the events described in Lemmas 3.1, 3.2, 3.3. By the induction hypothesis at the start of phase i , at least $n/2^i$ of the nodes, were in supervertices of size $\geq s(i)$, with prob $> 1 - n^{-2}$ and by Lemma 3.3 and previous results, all these nodes end up in acceptors with probability $\geq 1 - n^{-4}$. Suppose then that despite the conditions, less than $n/2^{i+1}$ nodes are in supervertices of size $\geq s(i+1)$. Then, at least $n/2^{i+1}$ of these nodes must be in supervertices of size $< s(i+1)$ i.e., $< \frac{s(i)^2}{2^i(1+\varepsilon)\log^4 n}$.

Call the set of these nodes V^1 . Let $\mathcal{A}_i$ be the number of acceptors at stage i . Then $\mathcal{A}_i \leq (1 - \varepsilon)(N_i) \frac{\log^K n}{\alpha s(i)}$ (this is so since, $N_i \cdot \frac{s(i)}{\beta \log n} \leq n$). So $\mathcal{A}_i \leq (1 - \varepsilon) \frac{n \log^{K+1} n}{\alpha s(i)^2} \beta$. By the Pidgeonhole principle, the size of some supervertex in V^1 must be at least $(\frac{n}{2^i}) / \mathcal{A}_i$

$$\geq \frac{\alpha s^2(i)}{(1 - \varepsilon) 2^i \log^{K+1} n \cdot \beta} > \frac{\alpha s^2(i)}{2^i (1 + \varepsilon) \log^4 n}.$$

But this is a contradiction to the definition of V^1 . □

Now let us assume that $N_i < s(i) \frac{(\beta - 1)}{\alpha}$ (this is the opposite of the condition of Lemma 3.2) and that there exists no large set of size greater than $\frac{n}{\log^4 n}$. Let $\alpha' = \sqrt{\alpha/(\beta(\beta - 1))}$. By applying the inequality

$N_i \geq \frac{n \log n}{2^i s(i)^\beta}$ we get $s(i) \geq \alpha' \sqrt[n \log n]{2^{-i}}$ i.e. $s(i) \geq \alpha' \sqrt{n} \cdot \log n$. We

can choose $\alpha > 1$ finally so that $\alpha' \geq 1$. The above result implies that the number of large supervertices gets very small only after the execution of stage 2. So, we conclude that:

LEMMA 3.5. *After the end of stage 2 with probability $\geq 1 - n^{-\gamma}$ for any $\gamma > 2$ we either have a supervertex which is bigger than $> \frac{n}{\log^4 n}$ vertices in size, or we have less than $\sqrt{n}$ supervertices, each of size at least $\sqrt{n} \cdot \log n$.*

Proof. By Lemmas 3.1, 3.2, 3.3, 3.4 and the previous remark and by taking products of probabilities of lemmas. $\square$

Now, we prove our full Lemma:

LEMMA 3.6. *At most one phase of stage 3 leaves $O(\log^4 n)$ supervertices around, with probability at least $1 - n^{-3/2}$. Then, the algorithm of stage 3 finds the connected components of the graph in $O(\log \log n)$ time.*

Proof. By Lemma 3.5, either we already have a very large supervertex of size at least $\frac{n}{\log^4 n}$ at the end of stage 2, or we have $\leq \sqrt{n}$ supervertices of sizes at least $\sqrt{n} \cdot \log n$ each, and the probability of the disjunction is $\geq 1 - n^{-a'}$. Stage 3 examines random edges with density $\frac{c'}{n}$. We now examine 2 cases:

Case 1. In the case of the existence of a large supervertex S of size

$> \frac{n}{\log^4 n}$, in just one phase, we claim that all but $\log^4 n$ of the remaining

vertices will join S with high probability. To prove this, we will use

Chernoff bounds on the Bernoulli experiment of $M \geq \frac{n}{\log^4 n}$ trials and independent

success probability at least $\frac{c'}{n}$. The probability that a particular vertex

does not join with S in one step, is at most

$$(1 - \frac{c'}{n})^{n/\log^4 n} = e^{-c' \log^4 n} = o(n^{-c'4}) .$$

The probability that more than $O(\log^4 n)$ vertices fail to join, is then

$$\log^4 n \cdot o(n^{-4c'}) = o(n^{-3c'}) = o(n^{-6})$$

Case 2. Consider a particular superverticex, A , of size at least $\sqrt{n} \cdot \log n$.

Consider another superverticex, say B , of size at least $\sqrt{n} \cdot \log n$. Let us

assume that B has at least one edge out of it. The probability that B 's edge does not connect to A is at most $(1 - \frac{\sqrt{n} \cdot \log n}{n})^{\sqrt{n} \cdot \log n} = (1 - \frac{\log n}{\sqrt{n}})^{\sqrt{n} \cdot \log n}$

$$\leq n^{-2} .$$

Hence, B will connect to A with probability at least $1 - n^{-2}$.

Since we have a total of $\leq \sqrt{n}$ of such large B 's, the probability that at least one of them will miss to hit A is at most $\sqrt{n} \cdot n^{-2} = n^{-3/2}$.

Hence, with probability at least $1 - n^{-3/2}$, all of the size of at least $\sqrt{n} \cdot \log n$ supervertices will hit A . This completes case 2.

In any of these two cases, the remaining vertices are at most $\log^4 n$ and it will take $O(\log \log n)$ parallel time for stage 3 to complete the finding of the connected components (with certainty, conditioned on the events of case 1 or 2). This completes the proof of Theorem 3.1. $\square$

3.2 Improving the Number of Processors

We now assume the P-RAM model with probabilistic choice instead of the W-RAM. In the P-RAM we can run MERGE in $O(\log n \cdot \log \log n)$ expected time, by simulating each probabilistic selection of a processor in a write-conflict by a tree of pairwise selections.

It is shown in [Chin, Lam, Len, 81] that the minimum of n elements can be computed on a P-RAM in time T where

$$T = \lceil n/k \rceil - 1 + \log k \quad \text{for } k \leq \frac{n}{2}, \quad k = \text{number of processors.}$$

Using the above technique, we have that the total expected time is

$$\sum_{t=0}^{\log \log n} \left(\frac{s(t)}{k} - 1 + \log k \right) \leq O\left(\frac{n}{k} + (\log \log n) \log k\right).$$

Selecting $k = n/(\log n \cdot \log \log n)$ we have the optimal value $O(\log n \cdot \log \log n)$ of the expected total time with an expected number of $nk = n^2/(\log n \cdot \log \log n)$ processors.

COROLLARY 3.2. *The expected parallel time for connected components of the random graph $G_{n,p}$ with $p \geq c/n$ is $O(\log n \cdot \log \log n)$ in the P-RAM model and the expected number of processors is $n^2/(\log n \cdot \log \log n)$.*

3.3 Biconnected Components

[Ja'Ja', 78] provided an algorithm for the P-RAM which finds the biconnected components of a graph in parallel time $O(\log n \cdot \log d \cdot \log k)$ where d is the depth of the graph and k is the number of biconnected components. The number of processors needed is $n^3/\log n$. Again, this algorithm can be implemented on the W-RAM model to run in $O(\log d \cdot \log k)$ parallel time with $n^3/\log n$ processors. The Appendix shows the expected depth is $\bar{d} = O(\log n)$. [Reif, Spirakis, 81] show the expected number of biconnected components is $\bar{k} = O(\log n)$. These results imply:

COROLLARY 3.3A. *The expected parallel time complexity for finding biconnected components in the random graph $G_{n,p}$ with $p \geq c/n$, $c > 5$, is $O((\log \log n)^2)$*

for the W-RAM model and $O(\log n \cdot (\log \log n)^2)$ for the P-RAM. The number of processors needed is n^3 for the W-RAM and $n^3 / \log n$ for the P-RAM. The probability that the parallel time is more than this is $o(\frac{1}{n})$.

By using Lemma 3.1 we get

COROLLARY 3.3B. The expected deterministic space for finding biconnected components in the random graph $G_{n,p}$ with $p \geq c/n$, $c > 5$, is $O(\log n \cdot (\log \log n)^2)$.

4. EXPECTED PARALLEL TIME FOR ALL PAIRS SHORTEST PATHS IN GRAPHS AND DIGRAPHS

Given an n vertex graph G or digraph D , the all pairs shortest path matrix A is an $n \times n$ matrix such that $A(i,j)$ is the length of a shortest path from i to j in G (or D). Let $A^k(i,j)$ denote the length of a shortest path from i to j going through at most k intermediate vertices. Clearly $A(i,j) = A^d(i,j)$ where $d = \text{depth of } G \text{ (or } D)$. Let $A^0(i,j)$ be 1 if $\{i,j\}$ is in G (if (i,j) is in D) and ∞ else. It is easy to see that

$$A^k(k,j) = \min_m \{A^{k/2}(i,m) + A^{k/2}(m,j)\}.$$

Hence we can compute A^d by computing $A^2, A^4, \dots, A^{2^i}$, $i = \lceil \log k \rceil$. A^k can be computed from $A^{k/2}$ using the matrix multiplication algorithm with $+$ substituted from $*$ and $\min$ for $+$. (See also [Dekel, Nassimi, Sahni, 81].)

Assuming binary representation of lengths and by using bit addition with parallel carry, we can perform all additions in the k^{th} phase of the algorithm in constant time on the W-RAM, by assigning one processor to each pair of (i,m) , (m,j) . The $\min$ operation can obviously be done in constant time in the W-RAM. To see this, we have to use $O(n^2)$ processors as follows: Assign $n-1$ processors to each m . Each one of them compares the computed $v_m = A^{k/2}(i,m) + A^{k/2}(m,j)$ with one of the $n-1$ other values. If v_m is bigger, then the corresponding processor writes a 1 in a memory location C_m , else it does not write. Next, one processor for each m checks C_m . If C_m is not written, then this processor outputs its v_m into $A^k(i,j)$. Only one C_m will not be written in this step. Hence:

THEOREM 4.1. *The all pairs shortest paths of a graph or digraph can be computed in $O(\log d)$ worst case time on a W-RAM, where d is the depth of the graph or digraph.*

Note that the above procedure is also valid if we are computing the all pairs *minimum weight* shortest paths provided that edge weights are nonnegative and satisfy the triangle inequality. The only modification needed is $A^0(i,j)$ = the weight of the edge (i,j) , if it exists, ∞ else. This leads to an expected time $O(\log \log n)$ for the W-RAM model for random graphs $G_{n,p}$ or digraphs $D_{n,p}$ since the expected d is $O(\log n)$ (see Appendix).

COROLLARY 4.1. *The expected time for all pairs shortest paths in $G_{n,p}$ or $D_{n,p}$ models with p as in the Theorems 1.1, 1.2 and arbitrary weights, non-negative and satisfying the triangle inequality, is $O(\log \log n)$ for the W-RAM model by using n^3 processors. It is $O(\log n \cdot \log \log n)$ in the P-RAM, by using $n^3 / \log n$ processors. The probability that the time is more than stated above, is $\leq n^{-\gamma}$, $\gamma > 2$.*

By Lemma 3.1 we get:

COROLLARY 4.2. *The expected deterministic sequential space for all pairs shortest paths in the random graphs $G_{n,p}$ or random digraphs $D_{n,p}$ (which are arbitrarily weighted with nonnegative weights satisfying the triangle inequality) is $O(\log n \cdot \log \log n)$.*

5. EXPECTED PARALLEL TIME FOR MINIMUM WEIGHT SPANNING TREE

5.1 An Algorithm for Minimum Spanning Trees of Undirected Graphs, in the W-RAM

Let G be a weighted undirected connected graph, with positive weights. The connectivity algorithm of [Hirschberg, Chandra, Sarwate, 79] can be used to obtain a minimum weight spanning tree, by picking the edge with minimum weight while merging supervertices with one another (see [Savage, 78]). Let G be an instance of the random graph $G_{n,p}$, $p \geq c/n$, as input, with arbitrary nonnegative edge weights. We can use our algorithm MERGE for connectivity, by using the [Ja'Ja', 78] algorithm for the deterministic part. Picking the minimal weight edge will take constant time in the W-RAM if we use $n^3 \cdot p$ processors. Hence,

COROLLARY 5.1. *Suppose we are given an instance of the random graph $G_{n,p}$, $p \geq c/n$ (c is the constant defined in 3.1), as input, with arbitrary nonnegative edge weights. Then we can compute a minimum weight spanning tree in the W-RAM in expected parallel time $O(\log \log n)$, using $n^3 p$ processors on the average.*

COROLLARY 5.2. *The expected deterministic sequential space for a minimum weight spanning tree is $O(\log n \cdot \log \log n)$, when the input graph is as in Corollary 5.1.*

5.2 An Alternative Method for Minimum Spanning Forests in the W-Ram, which Can Also Be Used for Directed Graphs

A parallel modification of the Kruskal algorithm will be used. First, we sort the edges of the graph (or digraph) into a sequence of nondecreasing weights. Then, separately for each edge, we test whether the endpoints of the edge belong to the same component of the graph (resp. strong component of the digraph) generated by edges of less than or equal weight. The edge will be incorporated into the constructed forest iff the answer is negative (i.e., if the endpoints of the edge belong to different components) (see [Kucera, 81], [Reif, 82]).

In the case of digraphs, the above procedure will produce actually a minimum weight directed acyclic graph (DAG) which spans the digraph.

The above technique has a time complexity bounded by the sorting, since the connectivity tests can be done in $O(\log \log n)$ expected time in the W-RAM, by employing the [Ja'Ja', 78] algorithm. Sorting takes $O(\log \log n)$ time in a W-RAM (see [Borodin, Hopcroft, 82] and [Shiloah, Vishkin, 80]).

LEMMA 5.1. *A minimum spanning DAG (directed acyclic graph) of a digraph can be found in $O(\log \log n)$ expected parallel time in the W-RAM.*

6. EXPECTED PARALLEL TIME FOR RANDOM GRAPH ISOMORPHISM

The following algorithm, given by [Babai, Kucera, 79] is essentially a Breadth-First-Search procedure applied to random graphs of the model $G_{n,p}$ with $p = 1/2$.

[1] Classify vertex by valences so each vertex is labelled by its valence. Let $c_1, \dots, c_n$ be the classes of the above classification, arranged by the order induced by this classification.

[2] In the first refinement we let $N_i(v)$ be the number of neighbors of v in c_i and let $N_i^4(v) =$ smallest nonnegative integer congruent to $N_i(v) \bmod 4$. Then, two vertices v, u are ordered now if (a) they were of different label before or (b) they had same labels but

$$(N_1^4(v), \dots, N_h^4(v)) < (N_1^4(u), \dots, N_h^4(u))$$

lexicographically.

[3] Let $c_1, c_2, \dots, c_h$ be the equivalence classes of [2]. Apply step [2] to each of these classes.

THEOREM 6.1. [Babai, Kucera, 79]. *Let U be the set of vertices whose class is not a singleton after step [3]. Then*

$$\text{Prob}\{|U| \geq 2\} \leq \exp(-2cn)$$

for some absolute constant $c > 0$.

The valence classification can be done in $O(\log \log n)$ time on the W-RAM model by using $O(n^2)$ processors. In the P-RAM, it can be done in time $O(\log n)$ by using $n^2 / \log n$ processors. The partial orders created by the refinements are best stored as a function corresponding to them. These function computations cost not more than $O(\log \log n)$ time in W-RAM and $O(\log n)$ in the P-RAM, by

using n^2 and $n^2/\log n$ processors, respectively. So:

COROLLARY 6.2. *A canonical labelling algorithm on $G_{n,p}$, $p = 1/2$, takes expected parallel time $O(\log \log n)$ in the W-RAM model with n^2 processors and $O(\log n)$ time in the P-RAM with $n^2/\log n$ processors.*

By Lemma 3.1, we get

COROLLARY 6.3. *The expected sequential deterministic space needed, for a canonical labelling algorithm on $G_{n,p}$, $p = 1/2$, is $O(\log n \cdot \log \log n)$.*

Acknowledgements

We apologize to U. Vishkin and Y. Shiloach for stating in a previous version of this paper that their parallel connectivity algorithm could be easily derived from a previously known algorithm.

REFERENCES

- Angluin, D. and L.G. Valiant, "Fast Probabilistic Algorithms for Hamiltonian Paths and Matchings," *J. Comp. Syst. Sci.* 18 (1979), pp. 155-193.
- Babai, L. and L. Kucera, "Canonical Labelling of Graphs in Linear Average Time," 20th IEEE Symp. on Foundations of Computer Science, Puerto Rico, 1979, pp. 39-46.
- Babai, L., P. Erdős, S.M. Selkow, "Random Graph Isomorphism," *SIAM J. Comp.* 9(3), Aug. 1980, pp. 628-634.
- Borodin, A. and J.E. Hopcroft, "Routing, Merging and Sorting on Parallel Models of Computation," 14th Annual ACM Symposium on Theory of Computing, San Francisco, CA, 1982.
- Chin, F., J. Lam, and I. Chen, "Optimal Parallel Algorithms for the Connected Components Problem," to appear in *Comm. of Assoc. for Computing Machinery*.
- Chernoff, H., "A Measure of Asymptotic Efficiency for Tests of a Hypothesis Based on the Sum of Observations," *Annals of Math. Statistics*, Vol. 23, 1952.
- Cook, S., "Towards a Complexity Theory of Synchronous Parallel Computations," Specker Symposium on Logic and Algorithms, Zürich, Feb. 5-11, 1980.
- Dekel, E., D. Nassimi, S. Sahni, "Parallel Matrix and Graph Algorithms," *SIAM J. Comp.* 10(4), Nov. 1981.
- Dymond, P. and S.A. Cook, "Hardware Complexity and Parallel Computation," IEEE 21st Symposium on Foundations of Computer Science, 1980.
- Erdős, P. and A. Renyi, "On the Evolution of Random Graphs," *Pub. Math. Inst. Hung. Acad. Sci.* 5A, 1960, pp. 17-61.
- Fortune, S. and J. Wyllie, "Parallelism in Random Access Machines," Proc. 11th ACM Symp. on Theory of Computing, May, 1978.
- Goldschlager, L., "A Unified Approach to Models of Synchronous Parallel Machines", Proc. 11th Annual ACM Symp. on Theory of Computing, May, 1978.
- Goldschlager, L., "Synchronous Parallel Computation," Ph.D. Thesis, Univ. of Toronto, Dept. of C.S., Dec. 1977.
- Hirschberg, D., A. Chandra, D. Sarwate, "Computing Connected Components on Parallel Computers," *Commun. of the ACM* 22(8), August 1979.
- Ja'Ja', J., "Graph Connectivity Problems on Parallel Computers," TR GS-78-05, Dept. of Computer Science, Penn. State Univ., PA, 1978.
- Karp, R.M., "The Probabilistic Analysis of Combinatorial Search Algorithms," *Algorithms and Complexity: New Directions and Recent Results*, J.F. Traub, Ed. Acad. Press, New York, 1976, pp. 1-19.
- Karp, R.M. and M. Sipser, "Maximum Matchings in Sparse Random Graphs," 22nd Symposium on Foundations of Computer Science, Nashville, TN, Oct 1981, pp. 364-375.

- Karp, R.M. and R. Tarjan, "Linear Expected Time Algorithms for Connectivity Problems," Proc. of the 12th ACM Symp. on Theory of Computing, Los Angeles, CAL, 1980, pp. 368-377.
- Kucera, L., "Parallel Computation and Conflicts in Memory Access," *Information Processing Letters*, Vol. 14, No. 2, April 1982.
- Nath, D. and S.N. Maheshwari, "Parallel Algorithms for the Connected Components and Minimal Spanning Tree Problems," *Information Processing Letters*, Vol. 14, No. 2, April 1982.
- Reif, J., "Symmetric Complementation," 14th ACM Symp. on Theory of Computing, San Francisco, CAL, May 1982(A), pp. 201-214.
- Reif, J.H., "On the Power of Probabilistic Choice in Synchronous Parallel Computations," 9th Colloquium on Automata, Languages and Programming, Aarhus, Denmark, July 1982(B), pp. 442-456.
- Reif, J., and P. Spirakis, "Random Matroids," Proc. of the 12th ACM Symp. on Theory of Computing, Los Angeles, CAL, 1980, pp. 385-347, also rewritten as "Probabilistic Analysis of Random Extension-Rotation Algorithms," TR-28-81, Aiken Comp. Lab., Harvard University, 1981.
- Reif, J. and P. Spirakis, "k-Connectivity in Random Undirected Graphs," TR-19-81, Aiken Comp. Lab., Harvard University, 1981.
- Ruzzo, W., "On Uniform Circuit Complexity," Proc. 20th Annual Symposium on Foundations of Computer Science, Oct. 1979.
- Ruzzo, W., personal communication, 1982.
- Savage, C., "Fast, Efficient Parallel Algorithms for some Graph Problems," Tech. Rep. Comp. Studies, North Carolina State Univ., 1978.
- Savage, C. and J. Ja'Ja', "Fast, Efficient Parallel Algorithms for Some Graph Problems," *SIAM J. Comp.* 10(4), Nov. 1981.
- Savitch, W.J., "Relationships Between Nondeterministic and Deterministic Tape Complexities," *J. Comp. System Sciences* 4(2), 1970, pp. 177-192.
- Schnorr, C.P., "An Algorithm for Transitive Closure with Linear Expected Time," *SIAM J. Comp.* 7, 1978, pp. 127-133.
- Shamir, E., and E. Upfal, "N-Processor Graphs Distributedly Achieve Perfect Matchings in $O(\log^2 n)$ Beats," ACM SIGACT-SIGOPS Symp. on Principles of Distributed Computing, Ottawa, Canada, 1982.
- Shiloach, Y. and U. Vishkin, "Finding the Maximum, Merging and Sorting in a Parallel Computation Model," Dept of C.S., Technion, Israel, TR173, March 1980.
- Shiloach, Y. and U. Vishkin, "A $O(\log n)$ Parallel Connectivity Algorithms," to appear in *J. of Algorithms*, 1982.

Spirakis, P., "Probabilistic Algorithms, Algorithms with Random Inputs and Random Combinatorial Structures," Ph.D. Thesis, Harvard University, December 1981.

Spirakis, P., and J. Reif, "Strong k-Connectivity in Digraphs and Random Digraphs," TR-25-81, Aiken Comp. Lab., Harvard University, 1981.

Wyllie, J., "The Complexity of Parallel Computation," Ph.D. Thesis, Cornell University, 1979.

APPENDIX

THE DEPTH OF A RANDOM GRAPH $G_{n,p}$ AND OF A RANDOM DIGRAPH $D_{n,p}$

We prove here Theorems 1.1 and 1.2.

LEMMA A.0. *Theorem 1.1 implies Theorem 1.2.*

Proof. Let D be any instance of $D_{n,p}$. The depth of D cannot exceed the depth of the underlying undirected version G of D (which is produced from D by removing all edge directions). However, G is an instance of $G_{n,p'}$ with $p' = 2p - p^2$. Theorem 1.1 holds for $G_{n,p'}$ for p' in the range

$$R = [0, 1] - \left[\frac{c_0}{n}, \frac{2c_0}{n} - \left(\frac{c_0}{n} \right)^2 \right]$$

where c_0 is a fixed constant (to be determined in the proof of Theorem 1.1). Hence, Theorem 1.2 holds for $D_{n,p}$ for p determined by $p' = 2p - p^2 \in R$. $\square$

A 1. Proof of Theorem 1.1 for $p = \theta(\frac{1}{n})$

We prove here Theorem 1.1 for undirected graphs. This provides an upper bound on both the average depth and the depth distribution of $G_{n,p}$ for all $p = \theta(\frac{1}{n})$ except those p in $\left[\frac{c_0}{n}, \frac{2c_0}{n} - \left(\frac{c_0}{n} \right)^2 \right]$ where c_0 is the cutoff constant for paths of length $d \log n$, $d > 2$. This provides a partial answer to the problem of finding the depth of $G_{n,p}$ for $p = \theta(\frac{1}{n})$, open up to now. The remaining cases of $p = o(\frac{1}{n})$ and $p = \Omega(\frac{1}{n})$ can be derived by results of [Erdős, Renyi, 59] (see A.2 and A.3 of the Appendix).

Consider the following two-stage experiment: Draw an instance of G_{n,p_1} with $p_1 = \frac{c'}{n}$ ($c' > 0$). Then, for each edge which failed to appear, draw again with probability $p_2 = \frac{c'}{n}$. The result of the combined experiments is an instance of the graph $G_{n,p}$ with $p = p_1 + (1-p_1)p_2 = \frac{2c'}{n} - \left(\frac{c'}{n} \right)^2 = \theta(\frac{c'}{n})$. The first stage of the above experiment constructs an instance I of G_{n,p_1} . Let $\mathcal{E}$ be the event "the instance I has a path of length $\delta \cdot \log n$ (where

δ is a fixed constant)" and $\bar{\mathcal{E}}$ be the complement of $\mathcal{E}$. Let us consider the probability of $\mathcal{E}$. We shall need the following Lemma:

LEMMA A1.1. ([Erdős, Renyi, 59]). Given $G_{n,p}$ with $p = \frac{c}{n}$, let $\tau_{\delta \cdot \log n}$ be the number of paths of length $\delta \cdot \log n$ in $G_{n,p}$. Then

$$\lim_{n \rightarrow \infty} \text{Prob}\{\tau_{\delta \cdot \log n} = 0\} = e^{-\lambda}$$

with

$$\lambda = (2c)^{\delta \cdot \log n - 1} \frac{(\delta \cdot \log n)^{\delta \cdot \log n - 2}}{\delta \cdot \log n!}.$$

The above lemma can be used to prove that " $\exists c_0 > 0$: for $c \geq c_0$ the graph $G_{n,p}$ with $p = \frac{c}{n}$ has at least one path of length $\delta \cdot \log n$ with probability $\rightarrow 1$ as $n \rightarrow \infty$ and for $c < c_0$ the $G_{n,p}$ with $p = \frac{c}{n}$ has no such path with probability $\rightarrow 1$ as $n \rightarrow \infty$ " (to prove that, find the c_0 such that $\lambda(c) \rightarrow \infty$ if $c \geq c_0$ and $\lambda(c) \rightarrow 0$ if $c < c_0$ for $n \rightarrow \infty$). Assume in our construction that $c' \geq c_0$. Then, $\text{Prob}(\mathcal{E}) \geq 1 - n^{-\alpha'(c')}$, where $\alpha'(c') > 2$ is a constant depending on c' . Let us condition the following on the event $\mathcal{E}$.

We start from the vertices of the path T of length $\delta \cdot \log n$ (at least such a path exists, by $\mathcal{E}$) and use them as the initial border set B_0 for a breath-first-search (BFS) process in the graph induced on the vertices of $G_{n,p}$ by just the second stage of the experiment.

Let $|B_0| = \delta \cdot \log n$ and let us find the distribution of the number of vertices to which the vertices of T are immediate neighbors. The set of these vertices will be the new border set of the second stage of BFS, and let us call it B_1 . We are interested in the $\text{Prob}\{|B_1| \geq 2 \cdot \delta \cdot \log n\}$. Let $|S_0|$ be equal to $|B_0| = \delta \cdot \log n$ (the number of visited vertices at the beginning of the first stage of BFS). The unvisited vertices are $n - |S_0| = n - \delta \cdot \log n$ and each of them has probability $h_0 = 1 - (1-p_2)^{\delta \cdot \log n}$ to be connected to B_0 . Hence

$$\text{Prob}\{|B_1|=x\} = \binom{n-|S_0|}{x} h_0^x (1-h_0)^{n-\delta \log n - x}$$

But $h_0 = 1 - \left(1 - \frac{c'}{n}\right)^{\delta \log n}$ and since $\frac{\log n}{n} \rightarrow 0$ as $n \rightarrow \infty$ we have that

$h_0 \rightarrow \frac{c' \delta \log n}{n}$ as $n \rightarrow \infty$. The mean number of $|B_1|$ is

$$(n-|S_0|)h_0 = c' \delta \log n - \frac{c'^2 \delta^2 \log^2 n}{n} \rightarrow c' \delta \log n \quad \text{as } n \rightarrow \infty.$$

Let $c' > 2$ and let us find the probability $\text{Prob}\{|B_1| < 2\delta \log n = 2|B_0|\}$.

From [Feller, 57] pg. 140 this is less than

$$Q = \binom{n-|S_0|}{2\delta \log n} h_0^{2\delta \log n} (1-h_0)^{n-\delta \log n - 2\delta \log n} \cdot r$$

where

$$r = \frac{(n-\delta \log n - 2\delta \log n + 1)h_0}{(n-\delta \log n + 1)h_0 - 2\delta \log n} = \frac{c' \delta \log n}{c' \delta \log n - 2\delta \log n} \rightarrow \frac{c'}{c'-2} \quad \text{as } n \rightarrow \infty.$$

Then Q is asymptotically equal to

$$\begin{aligned} & \binom{n-\delta \log n}{2\delta \log n} \left(\frac{c' \delta \log n}{n}\right)^{2\delta \log n} \left(1 - \frac{c' \delta \log n}{n}\right)^{n-3\delta \log n} \cdot \frac{c'}{c'-2} \\ & = n^{2\delta \log(c'/2)} n^{-c' \delta} \cdot \frac{c'}{c'-2} \leq n^{-\delta(c'-2\log(c'/2))} = n^{-\beta}, \end{aligned}$$

for $\beta = \delta(c'-2\log(c'/2))$.

Hence, $\text{Prob}\{|B_1| \geq 2 \cdot |B_0|\} \geq 1 - n^{-\beta}$. We now inductively form sets S_k and B_k such that $S_k = B_0 + B_1 + \dots + B_k$ and $h_k = 1 - (1-p_2)^{n-S_k}$ and B_{k+1} being selected according to the density

$$\text{Prob}\{B_{k+1}=x\} = \binom{n-S_k}{x} h_k^x (1-h_k)^{n-S_k-x}.$$

(This models the BFS process).

We claim that, as soon as $|S_k| \leq \frac{n}{2}$,

$$\text{Prob}\{|B_1| \geq 2 \cdot |B_{i-1}| \mid \text{given } F\} \geq 1 - n^{-\delta}$$

for

$$F = \bigwedge_{j=1}^{i-1} (|B_j| \geq 2|B_{j-1}|).$$

This is so, since the first extension (from the smallest border set $B_0 = \emptyset \cdot \log n$) is the most difficult. Then, the event

$$\mathcal{E}_1 = \bigwedge_{i=1}^{k+1} (|B_i| \geq 2|B_{i-1}|)$$

has probability $\geq \prod_{i=1}^{k+1} (1 - n^{-\delta}) \geq 1 - n^{-\delta+1}$ and let k be the last index for which $|S_k| \leq \frac{n}{2}$. Set $|S_k| = \frac{n}{2} (1 - \varepsilon)$ with $|B_k| \geq \frac{|S_k|}{2} = \frac{n}{4} (1 - \varepsilon)$. From $|S_k| = |B_0| + \dots + |B_k| \geq |B_0| + 2|B_0| + \dots + 2^k |B_0|$ we get $|S_k| \geq (2^{k+1} - 1) |B_0|$, implying

$$\begin{aligned} k &= \log \left(\frac{|S_k|}{|B_0|} + 1 \right) - 1 \\ &= \log \left(\frac{\frac{n}{2} (1 - \varepsilon)}{2^{\delta \log n}} + 1 \right) - 1 \\ &= \log n + \log \frac{1 - \varepsilon}{2^{\delta}} - \log \log n - 1. \end{aligned}$$

Hence

$$\begin{aligned} S_{k+1} &= S_k + B_{k+1} \geq S_k + 2B_k \geq \frac{n}{2} (1 - \varepsilon) + \frac{n}{2} (1 - \varepsilon) \\ &\geq n(1 - \varepsilon). \end{aligned}$$

So, we proved that:

LEMMA A 1.2. *Conditioned on the event $\mathcal{E}$, the EFS process will visit at least $n(1 - \varepsilon)$ nodes of G_{n,p_2} in τ_1 stages where $\tau_1 \leq \log n$, with probability $\geq 1 - n^{-\beta+1}$.*

Let us allow the BFS to continue until it can visit no more vertices, and let f be the final stage (such that $|B_{f+1}| = 0$). Consider the stage $f' \leq f$ such that $|S_{f'}| \leq n - \log n$ but either $|S_{f'+1}| > n - \log n$ or $|B_{f'+1}| = 0$. After f' either the process stops or there are at most $\delta \cdot \log n$ nodes more to visit and this cannot take more than $\delta \cdot \log n$ stages in either case. At f' , the set $B_{f'}$ of "unvisited" nodes is at least of size $\delta \cdot \log n$. Consider the backward process in which the set of unvisited nodes increases as stages are executed in reverse. The new process is symmetric to the BFS and because of the symmetry we can argue that:

LEMMA A1.3. With probability at least $1 - n^{-\beta+1}$, the process of backward BFS will reach a set of at least $n(1-\delta)$ nodes in number of stages $\tau_2 \leq \log n$.

By Lemmas A1.2 and A1.3, there is a stage number τ' such that $f' - \tau_2 \leq \tau' \leq \tau_1 \leq f$ in which the border sets of the two processes either coincide or one is a subset of the other. Hence, $\tau' + \tau_1$ is an upper bound to number of stages up to stage f and the total number of stages up to the end of the (forward) BFS is $\tau' + \tau_1 + \delta \log n \leq (2+\delta) \log n$. The probability of both the events considered in Lemmas A1.2 and A1.3 together is at least $1 - 2n^{-\beta+1}$. Hence, we have proved that

LEMMA A1.4. The diameter of the larger component (and hence the depth of) $G_{n,p}$ is $\leq (2+\delta) \cdot \log n$ with (conditional on $\mathcal{E}$) probability at least $1 - 2n^{-\beta+1}$.

Dropping condition $\mathcal{E}$, we get that:

The depth of $G_{n,p}$ with $p \geq \frac{2c'}{n} - \left(\frac{c'}{n}\right)^2$ is $\leq (2+\delta) \log n$ with probability at least $1 - \max(2n^{-\beta+1}, n^{-\alpha(c')})$, where $\beta = \beta(c' - 2\log(c'/2))$.

By selecting δ and c' suitably, we can guarantee a probability of at least $1 - n^{-\gamma}$, with $\gamma > 2$.

Note that for $p \leq c'/n$, Lemma A1.4 implies $\text{Prob}\{\text{depth of } G_{n,p} \leq \delta \cdot \log n\} \rightarrow 1$ as $n \rightarrow \infty$. Hence, our result holds for any p , $p \in [\frac{c'}{n}, \frac{2c'}{n} - (\frac{c'}{n})^2]$, where c' is the cutoff constant for paths of length $\delta \cdot \log n$, $\delta > 2$. This proves Theorem 1.1 with $c_0 = c'$ for the case $p = o(\frac{1}{n})$. $\square$

A2. The Case of $p = o(n^{-1})$

LEMMA A.2 ([Erdős, Renyi, 59]). Let the average valence of a graph of n vertices and m edges be the number $2m/n$. Let a graph be called balanced if its average valence is greater than, or equal to the average valence of any of its subgraphs. Then, the probability that $G_{n,p}$ contains at least one member of the class $B_{k,\ell}$ of balanced graphs of k vertices and ℓ edges is

$$\text{Prob}\{|B_{k,\ell}| \geq 1\} = O\left(\frac{p}{n^{-k/\ell}}\right).$$

COROLLARY A.2. $\text{Prob}\{\text{depth of } G_{n,p} < \log n\} \rightarrow 1$ as $n \rightarrow \infty$ for $p = o(n^{-1})$.

Proof. For $p = o(n^{-1})$ we can always write p as $p = (c/(n \cdot \omega(n)))$ where $c > 0$ is a constant and $\omega(n) \rightarrow \infty$ as $n \rightarrow \infty$. Let $B = B_{\log n, \log n - 1}$ be the class of paths of $\log n$ vertices. From Lemma A.4 we get

$$\begin{aligned} & \text{Prob}\{G_{n,p} \text{ contains at least one member of } B\} \\ &= O\left(\frac{c}{n \cdot \omega(n) \cdot n^{-\log n / (\log n - 1)}}\right) = O\left(\frac{c}{\omega(n)}\right) \rightarrow 0 \quad \text{as } n \rightarrow \infty \quad \square \end{aligned}$$

The above implies that the average depth of $G_{n,p}$ is less than $\log n$ for $p = o(n^{-1})$ and all $n > n_0$ for some n_0 .

A.3 The Case of $p = \Omega(n^{-1})$

LEMMA A.3. ([Erdős, Renyi, 59]). Let $p = (c/n) \cdot \omega(n)$ where $c > 0$ is a constant and $\omega(n) \rightarrow \infty$ as $n \rightarrow \infty$. Then, if $|B_{k,l}|$ is the number of balanced graphs of the class $B_{k,l}$ (k vertices, l edges) contained in $G_{n,p}$ we have

$$\text{Prob} \left\{ |B_{k,l}| < \omega(n)^2 \right\} = O\left(\frac{1}{\omega(n)}\right)$$

implying

$$\text{Prob}(|B_{k,l}| = 0) = O\left(\left(\frac{1}{\omega(n)}\right)^i\right).$$

COROLLARY A.3. The depth of $G_{n,p}$ is $\leq \log n$ for $p = \Omega(n^{-1})$, with probability going to 1 as $n \rightarrow \infty$. This implies that, for $p = \Omega(n^{-1})$, the average depth of $G_{n,p}$ is not more than $\log n$ for all $n > n_0$, for some $n_0 > 0$.

Proof. Let $B_{k,l}$ be the class of binary balanced trees of n vertices, $n-1$ edges. Then

$$\text{Prob}\{|B_{k,l}| \geq 1\} \geq 1 - O\left(\left(\frac{1}{\omega(n)}\right)^{n-1}\right).$$

Hence

$$\text{Prob}\{|B_{k,l}| \geq 1\} \rightarrow 1 \quad \text{as } n \rightarrow \infty. \quad \square$$

A.4 The Case of High Densities, $p \geq cn^{-1/3}$, $c > 1$

LEMMA A.4. The depth of $G_{n,p}$, for $p \geq cn^{-1/3}$ is ≤ 3 , with probability tending to 1 as n goes to ∞ . Here $c > 1$ is a constant.

Proof. Consider any pair u, v of vertices of $G_{n,p}$. Let S_u (respectively S_v) be the sets of neighbors of u (respectively of v).

Then, for any $\beta \in (0,1)$

$$\text{Prob}\{|S_u| \geq p(n-1)(1-\beta)\} \geq 1 - e^{-\beta^2(n-1)/2},$$

by the Chernoff bounds [Chernoff, 52]. For $\beta = 1/2$ we get

$$\text{Prob}\left\{|S_u| \geq \frac{cn^{2/3}}{2}\right\} \geq 1 - e^{-(c/8)n^{2/3}}.$$

Similarly,

$$\text{Prob}\left\{|S_v| \geq \frac{cn^{2/3}}{2}\right\} \geq 1 - e^{-(c/8)n^{2/3}}.$$

Hence

$$\text{Prob}\left\{\text{both } |S_u|, |S_v| \geq \frac{cn^{2/3}}{2}\right\} \geq 1 - 2e^{-(c/8)n^{2/3}}.$$

Let $\mathcal{E}$ be the event " $\text{both } |S_u|, |S_v| \geq \frac{cn^{2/3}}{2}$ ". Then,

$$\begin{aligned} \text{Prob}\{\text{no edge between } S_u, S_v \text{ given } \mathcal{E}\} &\leq (1-p)^{|S_u| \cdot |S_v|} \leq \left(1 - \frac{c}{3\sqrt{n}}\right)^{\frac{c^2}{4}n^{4/3}} \\ &= e^{-(c^3/4)n}. \end{aligned}$$

Hence, combining terms,

$$\text{Prob}\{\exists \text{ path of length } \leq 3 \text{ between } u, v\}$$

$$\geq \left(1 - 2e^{-(c/8)n^{2/3}}\right) \left(1 - e^{-(c^3/4)n}\right) \geq 1 - 3e^{-(c/8)n^{2/3}}.$$

Hence, $\text{Prob}\{\forall \text{ pairs } u, v \exists \text{ path of length } \leq 3 \text{ between them}\}$

$$\geq 1 - 3n^2 e^{-(c/8)n^{2/3}} \rightarrow 1 \quad \text{as } n \rightarrow \infty.$$

□

c.1

c.1

AUTHOR

TITLE

BORROWER'S NAME

FEB. 27 1984

FOURTEEN DAYS

A fine will be charged for each day the book is kept overtime.

PRINTED IN U S A

